Enhancement to a Computational Tool for the Support of the Design of Mixed Technology Systems

A.C.R. da Silva[1], T.S. Almeida[1] and I.A. Grout[2]

[1]Department of Electrical Engineering, Universidade Estadual Paulista (UNESP - FEIS), Ilha Solteira, Sao Paulo, Brazil

[2]Department of Electronic and Computer Engineering, University of Limerick, Limerick, Ireland

Emails: acrsilva@dee.feis.unesp.br; tiagoalmeida@ieee.org; Ian.Grout@ul.ie

Abstract

This paper presents an enhancement to the previously developed MS²SV tool and the use of the tool in two mixed technology case study designs. MS²SV was created to perform the translation of mixed technology system models developed in MATLAB/Simulink into a structural or behavioral description in VHDL-AMS. Previously, the MS²SV only translated models available within the integrated *LIB MS²SV* library. The enhancement to the tool presented and used here is an improvement to the original tool so that it now allows the designer to create their own design model libraries for model translation. To illustrate the use of the tool, a rudder controller and the Buck converter employed in an unmanned aerial vehicle were used as study case designs. For comparison with the original model in MATLAB/Simulink, the VHDL-AMS code obtained by the translation was simulated using the Mentor Graphics SystemVision environment. The results show the usability of the tool, using the translation improvement described in this paper, is improved.

Keywords

Mixed Technology System; MS²SV; Rudder Controller; Simulink; UAV; Buck Converter; ESL

Introduction

Continued efforts within the microelectronics industry to reduce the costs and time associated with the design, manufacturing and testing of an IC (integrated circuit) has led to important technology advances which have resulted in the realization of more complex ASIC (application specific integrated circuit) designs for varied application areas such as the automotive industry, aerospace, control and telecommunications. However, the task of developing new designs is not a trivial activity, making it necessary to utilize process automation and computational tools that can solve, for example, the problem of working with design descriptions created in different software design and analysis tools that are often saved in incompatible design description formats [1-5].

In the same way, the design of complex VLSI (very large scale integration) and ULSI (ultra large scale integration) microelectronic systems increases the need to eliminate manual and repetitive operations in order to reduce design time and costs, motivating the development of suitable design automation tools. To realize the automation of a design process, it is important to understand the problems encountered when working with a specific design project and the required design project implementation processes. In addition, the automation of processes within a particular design project requires the rapid implementation and analysis of various algorithms [6]. To illustrate the aspects considered in this work:

1. A top-down methodology for microelectronic systems design was presented in [7] and [8], and the need to model design functions in an adequate hardware description language (HDL), verification through simulation and synthesis of the model in a target technology were discussed.

2. The importance of hardware description languages in various aspects of the design of electronic devices were highlighted in [7], [8] and [9], ranging from documentation through to simulation and synthesis of the physical circuit. The work of [9] has a particular focus on the VHDL-AMS.

3. A methodology for the translation of mixed signal electronic systems using the VHDL-AMS language description was described in [3]. The computational tool has the capability of

translating electronic circuit models specified in MATLAB/Simulink to VHDL-AMS description. In this paper is presented an improvement of computational tool described in [4-5].

In this paper, the tool named MS²SV (MATLAB/Simulink to SystemVision) tool is discussed and the development of an enhancement to the original tool structure is presented. This enhancement provides for an increase in the usability of the tool and the ability to create models which can be translated by the tool from a MATLAB/Simulink model into a structural or behavioral description in VHDL-AMS. This paper is organized as follows. Section 2 introduces the MS²SV tool and the enhancements to the tool to be presented in this paper. Section 3 discusses the synthesis of electronic systems and related research. Section 4 presents in more detail the rudder controller and in section 5, the Buck converter topology is presented. In section 6, the enhancements made to the MS²SV are presented and system implementations created in MATLAB/Simulink. In section 6 the simulation results are discussed for both case study designs. Section 7 provides conclusions to the work presented in the paper.

Ms²sv Enhancements

A mixed-signal system can be modelled as a graphical block diagram and translated to VHDL-AMS structural description as described in [5]. With this approach, it is possible to quickly compare and analyse the simulation results obtained from the two model descriptions. This particular feature is presented in this paper with reference to the MS²SV tool. Furthermore, the contribution presented here describes the potential for the designer to create their own custom design libraries. This enhancement makes the computational tool more flexible. The MS²SV tool works directly with the MATLAB/Simulink environment [11] using the visual (block diagram) representation as the target for translation to a design format that is readable by the SystemVision environment from Mentor Graphics [12].

To test the new version of MS²SV, the rudder system control and Buck converter of an UAV (unmanned aerial vehicle) available in [13] were chosen as suitable case study designs. These examples were chosen considering different energy domains such as electrical and mechanical. **Figure 1** illustrates all the UAV system highlighting the part of the system studied in this paper.

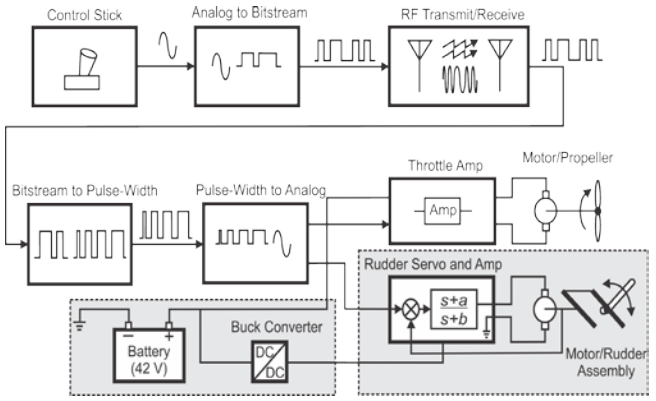


FIGURE 1 COMPLETE DESCRIPTION OF THE UAV SYSTEM, HIGHLIGHTING THE PARTS THAT WERE STUDIED IN THIS PAPER. MORE DETAILS CAN BE FOUND IN [13].

UAVs are guided without an on-board crew and they can fly autonomously or semi-autonomously. These craft are used mainly in the areas of surveillance and reconnaissance and they are indispensable in many applications where human intervention is impossible, very dangerous or expensive, e.g., hazardous material recovery, traffic monitoring, disaster relief support and military operations.

Related Research

Many papers concerning the synthesis of electronic systems consider the optimization or minimization of designs. For example, the computational complexity of the logic minimization problem at two levels in digital circuit designs was explored in [14]. The exploration of design complexity was considered in order to analyze the reduction of Boolean function by means of incorporating a particular circuit to a Boolean hypercube. A genetic algorithm for synthesis of multi-core architectures was developed in [15]. The target system was an implementation of a Petri net. After synthesis, the Petri net was implemented in software within a DSP (digital signal processor) for test purposes. The results shown the genetic algorithm is very robust for synthesis at the electronic system level (ESL).

Other steps involved in the optimization process of the design phases using widely known computational environments have also been considered. A GPS (Global Positioning System) design was developed using MATLAB/Simulink and presented in [16]. With the development of a based graphic design, it is also possible to quickly test and analyze complex designs.

Hardware description languages are widely used to design electronic systems and represent a means of simulation, synthesis and documentation of designs. For example, an algorithm was developed in [14] to automatically generate models of analog circuits for modeling faults at high level of abstraction. The algorithm was written in MATLAB and generated a description in VHDL-AMS for simulation and analysis of model failures using the SystemVision environment. A methodology was proposed for implementing FFT (fast Fourier transform) processors using VHDL and reconfigurable devices in [18]. A proposed architecture for implementing a SPICE (Simulation Program with Integrated Circuit Emphasis) simulator using the FPGA (field programmable gate array) was described in [19]. The architecture was implemented on an FPGA using the Verilog-AMS (Analog and Mixed Signal) language.

However, with so many studies involving different methodologies and computational tools, it was proposed in [20] that a classification model for designs would be beneficial. The paper detailed the characteristics and approaches that differs the computational tools of design, both for modeling hardware and software.

Figure 2 illustrates the design flow at different levels of abstraction from the high level (ESL) downwards to more detailed design descriptions which would be ultimately suitable for implementation. The design flow model is considered in [20] as an extended version of the Y diagram presented in [7]. The paper however pointed out the problems of incompatibility between the tools and weaknesses they have. **Figure 2** defines the process of designing the top-down methodology in an ideal model, called the Double Roof Model. One side (left) corresponds to the process of software creation, whilst the other side (right) corresponds to the process of hardware creation. Each side is divided into different levels of abstraction, for example, **Task** and **Instruction** (hardware) or **Components** and **Logic** (software). There is a common level of abstraction, called ESL (electronic system level). At this level, the distinction between hardware and software cannot be made. At each level, a synthesis step (continuous vertical arrow) can be run and the specification is transformed into an implementation. Horizontal dotted arrows indicate the step which can change the models of individual elements in the implementation directly to the next level of abstraction (lower level of abstraction). The proposed methodology focuses on the specification and implementation at the ESL *level*, since it is not possible to distinguish between an implementation in hardware or software.

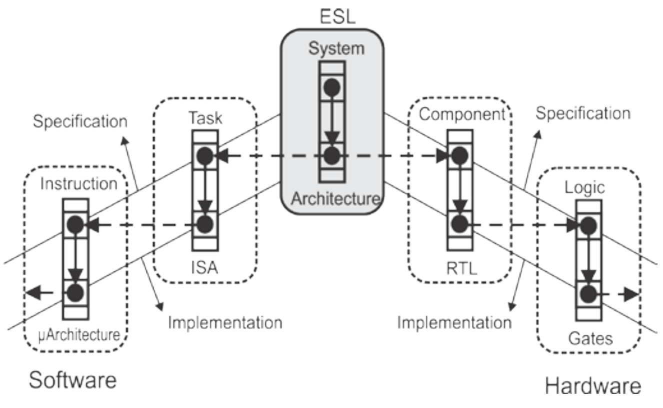


FIGURE 2 FLOW DESIGN OF ELECTRONIC SYSTEMS, ALSO KNOWN AS DOUBLE ROOF MODEL [20]

Specifications of the Rudder Control System

The controller proposed in this paper follows the specifications described by [11]. The design was developed according to five main specifications described in **Table I**.

TABLE 1 GENERAL SPECIFICATIONS FOR THE RUDDER CONTROL SYSTEM

FEATURES	VALUES
RUDDER DEFLECTION ANGLE	± 60° (± 1.05 RADIANS)
RUDDER SPEED (0 TO 60°)	≤ 300 MS
TORQUE AT GEARBOX	≥ 0.7 NM
SERVOLOOP ERROR	≤ 1%
SERVOLOOP PHASE MARGIN	≥ 35%

The system has been described and analyzed as a collection of blocks built as transfer functions modeled using Laplace transforms which represent a continuous time system. With this implementation, it is possible to determine information about the system operation such as the position of the servo motor and phase margin. Models at this high-level abstraction of a continuous system allow the developer to ignore implementation considerations such as target technology and the interfacing between the different energy domains. For example, it can disregard connections that convert electrical outputs to mechanical inputs as the model inputs and outputs are simply defined as real quantities (numbers) [9]. **Figure 3** shows the general functional diagram of the model generated for the rudder system controller. The model ignores translational linkage as this was not considered in this paper. The rudder control system is modeled as a controller closed loop system. The limiter guarantees that no more than ±4.8 V here is available to the motor (for the system implementation).

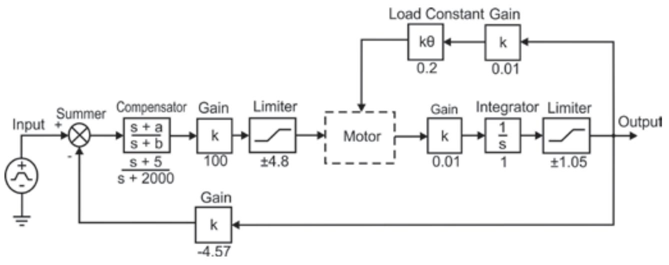


FIGURE 3 GENERAL FUNCTIONAL DIAGRAM OF THE SERVO LOOP WITH A BROADER VIEW OF THE BLOCKS THAT MAKE UP THE SYSTEM, INCLUDING VOLTAGE LIMITERS AND LAPLACE TRANSFER FUNCTION BLOCKS

The loop is closed by feeding back the output to the summer (sum) block. At the servo controller, these values have opposite signs so the summer operates by calculating the difference. There are two limiter blocks in the design. The first limiter is projected to limit the error signal overall for the level of supply voltage ±4.8 V. The second limiter is used to model the mechanical stop that prevents the gearbox shaft from rotating beyond a ± 60° arc. This block is configured to limit some signal to ± 1.05 radians, which corresponds to ± 60°. The compensator block is necessary to achieve the desired accuracy in the closed loop system without the system becoming unstable [13]. This difference between precision and stability is normal in closed loop systems and simulation technologies are very convenient to detail it. The gain is modeled simply as a gain block, with a value equal to 0.01. The purpose of the integral block is to convert angular velocity output of the gearbox as a proportional angular position. The blocks representing the servo loop are presented within the dotted box shown in **Figure 4**.

The implementation of the motor was chosen so that the load of torque (the result of the air force against the rudder) can be directly subtracted from the torque generated. This subtraction is engaged as a collection of blocks that isolate the torque generated by the motor. Once the generator torque is isolated, the sum is included as a contribution of external torque (positive or negative) to be added to the torque generated by motor.

The specifications necessary to the motor blocks are described in **Table II**.

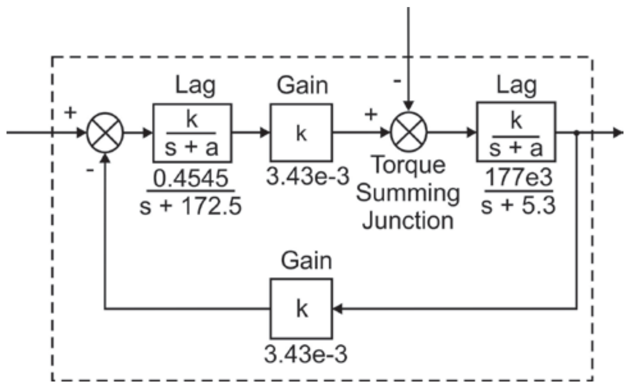


FIGURE 4 FUNCTIONAL DIAGRAM OF THE MOTOR USED IN THE RUDDER CONTROL. THE LAG BLOCKS ARE IMPLEMENTED AS LOW-PASS FILTERS

TABLE 2 CONFIGURATION VALUES FOR THE MOTOR BLOCKS USED IN RUDDER CONTROLLER MODEL

FEATURES	VALUES
ARMATURE WINDING RESISTANCE	R = 2.2 Ω
ARMATURE WINDING INDUCTANCE	L = 2.03 MH
SHAFT INERTIA	J = 168 KG/M^2
TORQUE (MOTOR) CONSTANT	K_T = 3.43 MNM/A
VOLTAGE (BACK-EMF) CONSTANT	K_E = 3.43 MV/RAD/S
VISCOUS DAMPING FACTOR	B = 5.63 μNM/RAD/S

The motor model is mainly composed of lag and gain blocks. Lag blocks represent the electrical time constant and the mechanical time constant, and the electrical gain block represents the voltage and constant torque. The lag block parameters can be determined using the following expressions:

$$\tau_e = \frac{L}{R} \tag{1}$$

$$\tau_m = \frac{J}{B} \tag{2}$$

where τ_e is the electrical time constant, and τ_m is the mechanical time constant. Since these time constants will be implemented using a model that accepts frequency rather than time parameters, the constant of time as pole positions can be expressed as:

$$p = -\frac{1}{\tau} \tag{3}$$

where *p* is the pole position in rad/s. To determine the actual pole values for this motor, the motor specifications along with (1), (2) and (3) were used. The electrical pole was calculated at 1.08 krad/s (172.5 Hz), and the mechanical pole was calculated at 33.5 rad/s (5.3 Hz). To complete the motor model definition, the values of the gain throughout the motor implementation also needed to be determined. The dedicated gain blocks were both set to 3.43 x 10^{-3}, which corresponds to the voltage and torque constants for the motor. The electrical lag block had a DC gain of 0.4545, which was calculated by solving the following transfer function with $s = 0$:

$$T(s)_e = \frac{1}{sL + R} \tag{4}$$

The mechanical lag block has a DC gain of 177.6 x 10^3, which was calculated by solving the following transfer function with $s = 0$:

$$T(s)_m = \frac{1}{sJ + B} \tag{5}$$

Specifications of the Buck Converter

Switching power supplies use the technique of pulse width modulation (PWM) in order to achieve efficiency and provide control of output voltage with changing load conditions. Because of the combination of electronic and magnetic components in a switching power supply, computer simulation plays a vital role in creating a successful design. All basic types of switching power supplies using PWM techniques have the same four basic elements: (i) a switch for setting the waveform PWM control, (ii) a diode, (iii) an inductor, and (iv) a capacitor. The cycle of the switch control signal determines how long the switch is closed for a given period and therefore can be used to control the amount of energy stored in the inductor. A complete switching power supply also has a transformer to provide isolation between input and output and feedback to control the cycle of the PWM waveform with load conditions changes.

The Buck converter model (for step down dc to dc conversion) considered here follows the specifications of Table III and **Figure 5** illustrates the simple model of Buck converter. The buck converter shown in **Figure 5** has two basic modes of operation: continuous mode and discontinuous mode, referring to the current flowing through the inductor (L1):

1. In continuous mode, current is always flowing through L1 whether the switch (SW1) is on or off.
2. In discontinuous mode the inductor current goes to zero during part of the off time of SW1.

TABLE 3 SYSTEM SPECIFICATIONS FOR THE UAV POWER SUPPLY

FEATURES	VALUES
V_{IN}	42 V CC
$F_{SWITCHING}$	25 KHZ
V_{OUT}	4.8 V CC
$V_{OUT(TORQUE)}$	< 100 MV P-P
I_{OUT}	15 MA A 2 A
$I_{OUT(TORQUE)}$	< 30 MA P-P

Since the frequency response is significantly different between the two modes, it is best to operate in only one of the modes [13]. The amount of energy transferred to the load is controlled by the duty cycle of the switch control waveform. The duty cycle (D) is defined as:

$$D = \frac{T_{on}}{T_s} \tag{6}$$

where T_s is the total switching period and T_{on} is the amount of time the switch is on. The duty cycle can range from 0.0 to 1.0, but typically falls between 0.05 and 0.95 (5% to 95%).

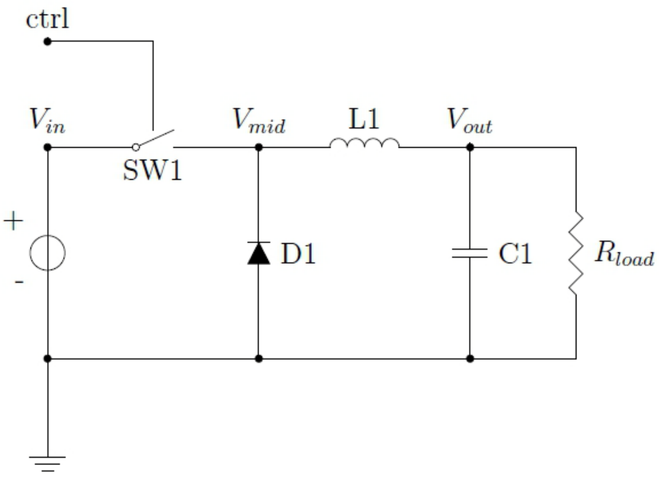


FIGURE 5 BASIC TOPOLOGY OF THE BUCK CONVERTER USED IN THE PAPER

From Equation (6), the equations for T_{on} and T_{off} can be derived:

$$T_{on} = D \times T_s \tag{7}$$

$$T_{off} = (1 - D) \times T_s \tag{8}$$

The first step in designing a switching regulator is to select the duty cycle. For the Buck converter operating in continuous mode, the following relationship can be used to approximate the duty cycle:

$$V_{out} = V_{in} \times D \tag{9}$$

From specifications, the duty cycle required to give the desired output can be calculated. However, since the desired output was relatively small (4.8 V), it needed to include the diode voltage drop in the calculation as follows:

$$V_{out} = (V_{in} \times D) - V_d \tag{10}$$

Solving for *D* in equation 10, the equation becomes:

$$D = \frac{V_{out} + V_d}{V_{in}} = \frac{4,8 - 0,7}{42} = 0,131 \tag{11}$$

From Equations (7) and (8), the on and off time of the control input could also be calculated. Recalling from the power supply requirements that the switching frequency, *f*$_s$, is 25 kHz then $T_s = 1/f_s = 40$ μs. Noting also $T_s = T_{on} + T_{off}$ and substituting in Equations (7) and (8):

$$T_{on} = 0,131 \times 40\mu s = 5,24\mu s \tag{12}$$

$$T_{off} = (1 - 0,131) \times 40\mu s = 34,76\mu s \tag{13}$$

The next step was to calculate the values for the inductor (L1) and capacitor (C1) in the output filter. For the inductor the following equation was used:

$$V_L = L \times \frac{dI_L}{dt} \approx L \times \frac{\Delta I_L}{\Delta t} \tag{14}$$

where V_L is the voltage across the inductor and I_L is the inductor current. The goal here is to select a minimum value for L1 such that the converter operates in continuous mode. To calculate the minimum inductance to remain in continuous mode, equation 14 can be rearranged in order to solve for L_{min}:

$$L_{\min} = V_L \times \frac{\Delta t}{\Delta I_{L(\max)}} \tag{15}$$

$$L_{\min} = (V_{in} - V_{out}) \times \frac{T_{on}}{\Delta I_{L(\max)}}$$

$$L_{\min} = (42 - 4,8) \times \frac{5,2\mu s}{30mA}$$

$$L_{\min} = 6,5mH$$

where T_{on} is the on time and $\Delta I_{L(\max)}$ is the maximum ripple current. Here, the equivalent circuit for the on state was used, neglecting the on resistance for the switch. The inductor value calculated guarantees continuous-mode operation as long as the load current does not exceed 15 mA. The capacitor controls the amount of ripple voltage on the output. The following formula can be used to calculate the minimum capacitance for a Buck converter:

$$C_{\min} = \frac{\Delta I_{out}}{8 \times f_s \times \Delta V_{out}} \tag{16}$$

$$C_{\min} = \frac{30mA}{8 \times 25kHz \times 100mV}$$

$$C_{\min} = 1,5\mu F$$

Using a capacitor value equal to or greater than this value guarantees the ripple voltage will be below 100 mV. The exact value for the capacitor is not critical and is often up to 10 times the minimum calculated value. The final step is to calculate the minimum and maximum load resistance. This could be determined from the load current and voltage specifications:

$$R_{load(\min)} = \frac{V_{out}}{I_{out(\max)}} = \frac{4,8V}{2A} = 2,4\Omega \tag{17}$$

$$R_{load(\max)} = \frac{V_{out}}{I_{out(\min)}} = \frac{4,8V}{15mA} = 320\Omega \tag{18}$$

The $R_{Load(max)}$ value is critical to ensure the current does not fall below 15 mA and force the circuit into the discontinuous mode.

Methods and Materials

The computational tool called MS^{2}SV developed in [4-5] is capable of generating a structural description in VHDL-AMS from a block diagram, as described in Section 1. The enhancement discussed in this paper allows the designer to create native Simulink libraries and to configure the new version of MS^{2}SV so that it recognizes the elements within the library in order to generate a description a VHDL-AMS based equivalent model. It can also be configured by the designer to recognize the Simulink MS^{2}SV primitive elements.

Description of the Computational Tool Development

For the MS^{2}SV to be set-up so that it can work to be configurable for different projects and systems, the improvements undertaken uses a file structure that contains the models recognized by the tool. There is a directory called **bin** where the files **lib_ref.ini** and

blk_ref.ini are located. These files reference libraries and elements of the Simulink toolbox which are recognized and used by MS²SV.

If there are subsystems, it will be generated as VHDL-AMS code for each subsystem used by the model so that design hierarchy is maintained. The MS²SV allows the relationship between elements in the same library by creating hierarchical models. The improvement was implemented using the C++ language and the object-oriented programming paradigm. Thus, **Figure 6** illustrates a class diagram MS²SV through a software perspective. Note that for space limitations in the paper, not all of the interface layer is represented as some aspects are not significantly influence the functionality of MS²SV.

In the diagram shown in **Figure 6**, only the classes **Main**, **Save** and **VHDL** are part of the user interface layer. All other classes are part of the translation engine translation rules. The class **SystemFile** is responsible for generating the project structure for simulation in SystemVision.

The class **TranslateCode** is responsible for generating the system description in VHDL-AMS. This class inherits the characteristics of the class **Checking**, which is responsible for verifying the existence of Simulink primitive, user libraries and subsystems. **Checking** inherits the characteristics of **Lib**, which is responsible for capturing the models contained in the library. Finally, **Lib** inherits the characteristics of **ReadMDL**, which is responsible for interpreting the model in Simulink.

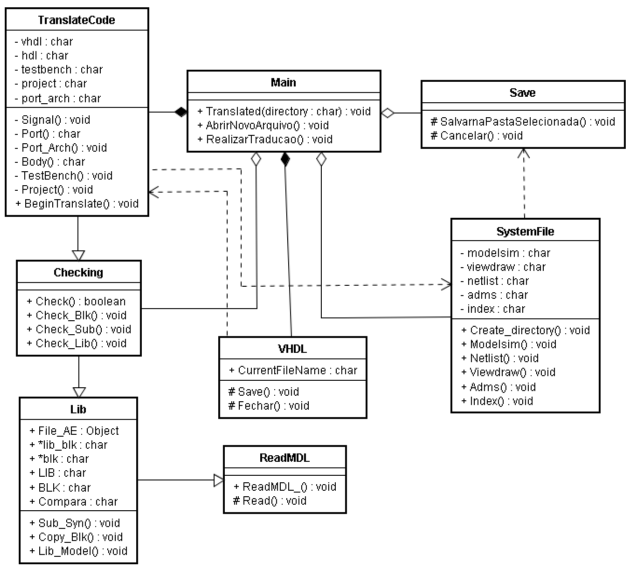


FIGURE 6 MS²SV CLASS DIAGRAM REPRESENTING THE RELATIONSHIPS OF INHERITANCE, DEPENDENCY, COMPOSITION AND COMPOSITION ON AGGREGATION

In **Figure 7** is illustrated the MS²SV sequence diagram for the case of translation of the primitive elements in the Simulink toolbox.

The translation process starts with a check of the elements present in the model by calling direct the operations of the class **Checking**. **Checking** class uses the characteristics of **ReadMDL**. Then, **TranslateCode** begins the process of translating and reading the file Simulink. Reading the file from Simulink is made more than once, because the user interface is controlling the translation rules engine, allowing loading various files without translation.

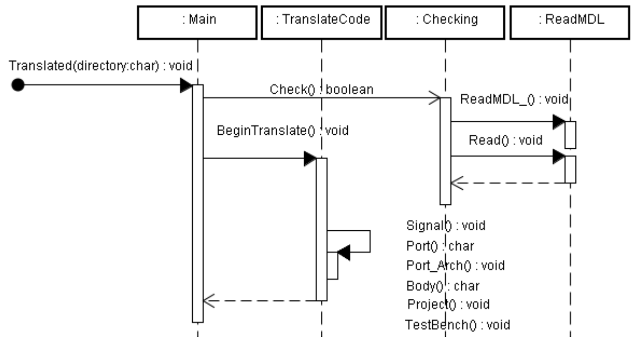


FIGURE 7 SEQUENCE DIAGRAM OF MS²SV REPRESENTING THE TRANSLATION PROCESS OF ELEMENTS BELONGING TO THE TOOLBOX OF SIMULINK PRIMITIVE, WITHOUT INTERACTION WITH USER LIBRARIES

Figure 8 illustrates the sequence diagram for the case of translation elements contained in libraries created by the designer. The process is quite similar to the diagram of **Figure 7**, bu t with the inclusion of the **Lib** class, which is responsible for transcribing the description already saved in the internal library MS²SV representing the model described in the library of the designer.

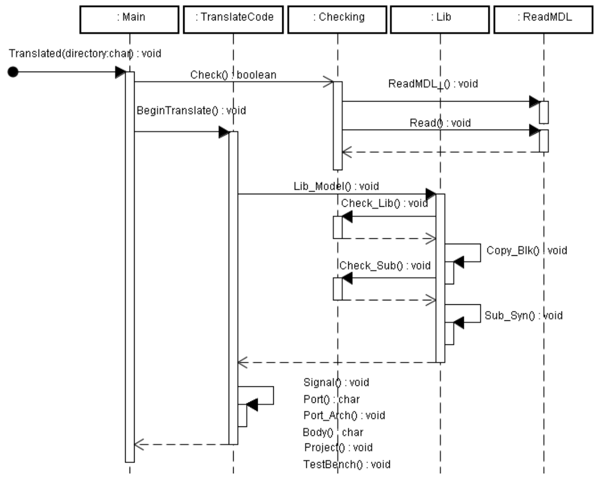


FIGURE 8 SEQUENCE DIAGRAM OF MS²SV REPRESENTING THE TRANSLATION PROCESS OF ELEMENTS BELONGING TO USER-CREATED LIBRARY OR EVEN IN CONJUNCTION WITH PRIMITIVE ELEMENTS

Simulink Modelling

To model the rudder controller in the MATLAB/Simulink environment, the standard components available in the toolbox were used since Simulink has basic elements for general purpose use - such as gain, integral and sum. For the generation of the input voltage, a sine generator block was used and the gain values were adjusted in accordance with the specifications illustrated in **Figure 3**. **Figure 9** illustrates the overview of modeling in Simulink. To model the compensator block, a subsystem in Simulink environment was created, necessitating the use of two different components; the transfer function using the Laplace transform component and a gain. **Figure 10** illustrates the construction of the block compensator.

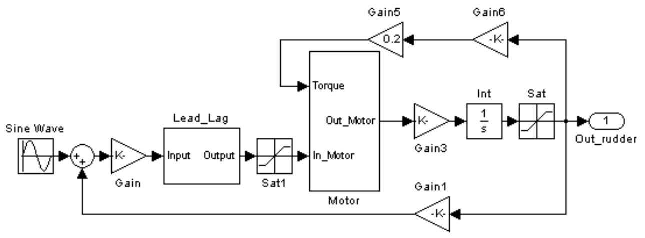


FIGURE 9 BLOCK DIAGRAM BUILT IN MATLAB/SIMULINK ENVIRONMENT DESCRIBING THE RUDDER CONTROLLER FOLLOWING THE SPECIFICATIONS OF THE MODEL

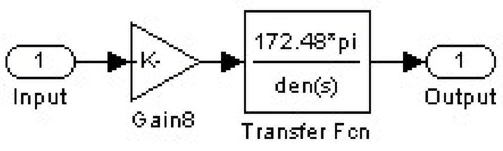


FIGURE 10 SUBSYSTEM CREATED TO MODEL THE BLOCK COMPENSATOR RESPONSIBLE FOR THE ACCURACY WISHED IN RUDDER CONTROLLER

For modeling the motor, as shown in **Figure 11**, the basic components sums, gains were used and two first-order low pass filters were also created. Since the filters also use the Laplace transfer function, they were built in the same way that the compensator shown in **Figure 10**, but with appropriate settings.

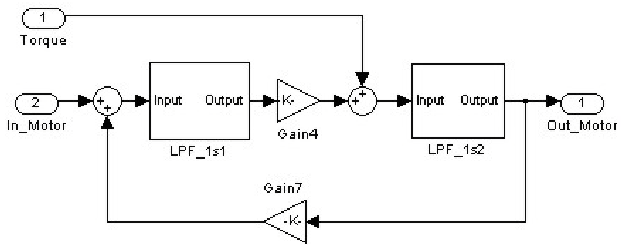


FIGURE 11 SUBSYSTEM CREATED FOR THE DC MOTOR EMPLOYED IN THE GENERATION OF TORQUE BY USING THE SUBSYSTEMS TO CREATE TWO LOW-PASS FILTERS OF FIRST ORDER

The description of the Buck converter obeyed the specifications identified in Section 5. However, this model did not use the Simulink primitives. As the modeling required the use of discrete elements (such as inductor, capacitor, resistor, etc.), it required the use of the **Simspace** library. **Figure 12** illustrates the entire system of the buck converter in MATLAB/Simulink.

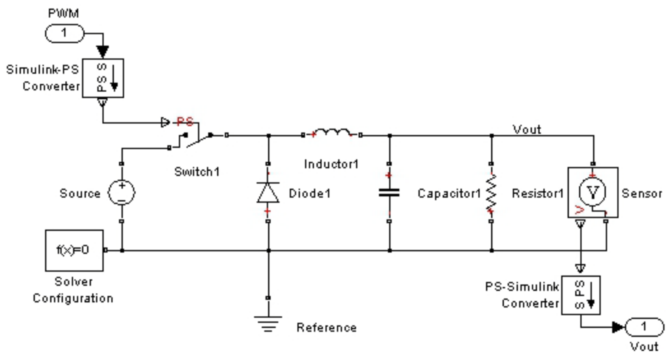


FIGURE 12 MODEL BUCK CONVERTER USING MATLAB/SIMULINK ENVIRONMENT

DC-DC converters are subject to sudden changes in conditions, such as variations in input voltage or changes in output load. These changes can cause the output voltage to vary outside the specified limits. However, this work did not consider a model able to correct the variation in supply voltage, so generating an ideal voltage at the output. After the creation of the models described, a library of Simulink call **LibRS_S_Domian** was created. In this library were added blocks for the compensator model, first-order low-pass filters, rudder controller, DC motor and Buck converter. From the **LibRS_S_Domian** the MS^{2}SV was configured to recognize this new library so that it was possible to translate the model.

Simulation and Results

To simulate the rudder controller in the Simulink environment, a small piece of MATLAB source code to plot graphs pertaining to the waveform of the input and output signals was required. With the simulation time of 1 second and a frequency of 1 Hz, it was possible to generate a sine wave representing the voltage ranging within ± 4.8 V. The output was captured and properly matched to ± 1.05 radians according to the torque command input voltage passed through. **Figure 13** illustrates the simulation result obtained in Simulink environment where a sine wave input is applied to the rudder servo.

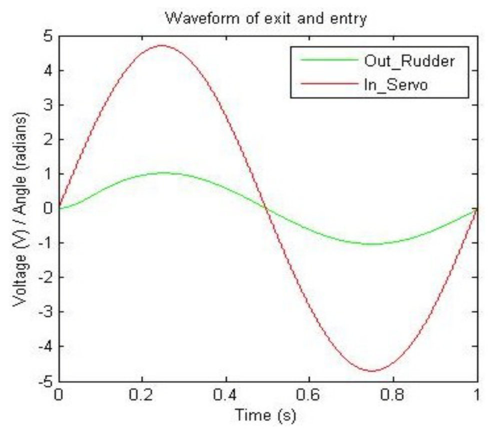


FIGURE 13 SIMULATION RESULT FOR THE INPUT AND OUTPUT OF RUDDER CONTROLLER IN THE MATLAB/SIMULINK ENVIRONMENT

The MS^{2}SV generated a combination of descriptions in structural VHDL-AMS, representing more clearly the links between the component parts of the system. An example of the structural description in VHDL-AMS generated for the motor is listed below.

```
--Generated by MS2SV version 1.7
--File created Mon Feb 25 14:59:13 2012

library IEEE;
use ieee.electrical_systems.all;
library EduLib;
use WORK.all;
library fundamentals_vda;
library spice2vhd;

entity DC_Motor is
port (terminal Tq: electrical;
    terminal In_: electrical;
    terminal Out_: electrical);
end entity DC_Motor;

architecture arch_DC_Motor of DC_Motor is
    terminal \1N1\: electrical;
    terminal \1N0\: electrical;
    terminal \1N3\: electrical;
    terminal \1N2\: electrical;
begin
E_Gain4: entity EDULIB.E_GAIN(BEHAVIORAL)
generic map ( K => 3.43e-3 )
port map ( INPUT => \1N0\,
    OUTPUT => \1N1\ );
E_GAIN7 : entity EDULIB.E_GAIN(BEHAVIORAL)
generic map ( K => 3.43E-3 )
port map ( INPUT => Out_,
    OUTPUT => \1N4\ );
E_LPF_1ST5 : entity WORK.LPF_1S1(S_DMN)
generic map ( FP => 172.48,
    K => 0.4545 )
port map ( INPUT => \1N3\,
    OUTPUT => \1N0\ );
E_LPF_1ST6 : entity WORK.LPF_1S2(S_DMN)
generic map ( FP => 5.33,
    K => 177.67E3 )
port map ( INPUT => \1N2\,
    OUTPUT => Out_ );
E_Sum1: entity EDULIB.E_SUM
port map ( IN2 => \1N4\,
    IN1 => In_,
    OUTPUT => \1N3\ );
E_Sum2: entity EDULIB.E_SUM
port map ( IN2 => \1N1\,
    IN1 => Tq,
    OUTPUT => \1N2\ );
end architecture arch_DC_Motor;
```

Importantly, the model required behavioral descriptions of each block mapped in the structural description. As MS^{2}SV generated all the project files and structure for SystemVision, there are project files to make reference to the internal library in SystemVision where the behavioral descriptions necessary for simulation can be found. After translation of the model, the simulation time was configured with 1 second in the SystemVision environment. All settings are inherited from the model described in Simulink, so the input voltage representing the command of the servo motor is the same. Hence, the output was the same ± 1.05 radians.

Figure 14 illustrates a waveform obtained with the simulated using the SystemVision environment and this can be compared to the simulation results for the same stimulus obtained from the Simulink environment as shown in **Figure 13**.

To simulate the Buck converter, a power source of 42 V was used and this was converted correctly to 4.8 V, as shown in **Figure 15**. The simulation was performed in one step of 20 ms duration.

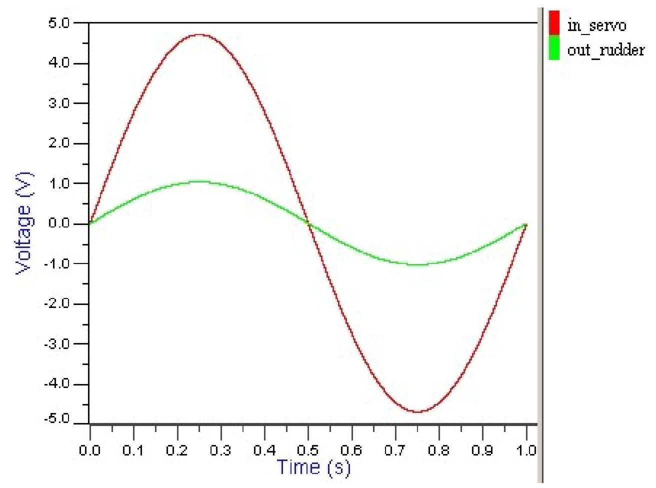


FIGURE 14 WAVEFORM SIMULATION OF THE INPUT AND OUTPUT SIGNAL WITH VHDL-AMS DESCRIPTIONS OF RUDDER CONTROLLER IN THE SYSTEMVISION ENVIRONMENT

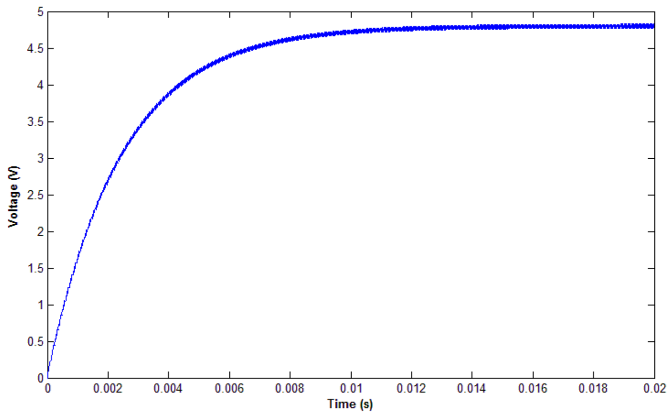


FIGURE 15 WAVEFORM SIMULATION OF THE PROPOSED BUCK CONVERTER IN MATLAB/SIMULINK ENVIRONMENT

The same approach was used for the rudder controller case study. Below is a list of description in VHDL-AMS generated by MS^{2}SV. Also, behavioral descriptions are referenced in the internal library in SystemVision.

```
-- Generated by MS2SV version 1.7
-- File created Mon Aug 06 16:08:12 2012

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.electrical_systems.all;
USE ieee.mechanical_systems.all;
USE ieee.fluidic_systems.all;
USE ieee.thermal_systems.all;
USE ieee.radiant_systems.all;
LIBRARY edulib;
USE work.all;

entity BUCK is
  port (
      terminal VOUT: electrical;
      signal CTRL_D: in std_logic
  );
end entity BUCK;

architecture arch_BUCKCONVERTER of BUCK is
   terminal VMID: ELECTRICAL;
   terminal VIN: ELECTRICAL;
 Begin
   SW1 : entity EDULIB.SWITCH_DIG
   port map ( SW_STATE => CTRL_D,
      P1 => VIN,
      P2 => VMID
   );
   D1 : entity EDULIB.DIODE(IDEAL)
   port map ( P => ELECTRICAL_REF,
      N => VMID
   );
   RLOAD : entity EDULIB.RESISTOR(IDEAL)
   generic map ( RES => 2.4 )
   port map ( P1 => VOUT,
      P2 => ELECTRICAL_REF
   );
   VINDC : entity EDULIB.V_CONSTANT(IDEAL)
   generic map ( LEVEL => 42.0 )
   port map ( POS => VIN,
      NEG => ELECTRICAL_REF
   );

   C1 : entity EDULIB.CAPACITOR(IDEAL)
   generic map ( CAP => 1.5E-6 )
   port map ( P1 => VOUT,
      P2 => ELECTRICAL_REF
   );

   L1 : entity EDULIB.INDUCTOR(IDEAL)
   generic map ( IND => 6.5E-3 )
   port map ( P1 => VMID,
54.         P2 => VOUT
55.     );
56.  end architecture arch_BUCKCONVERTER;
```

The parameterisable component values identified in lines 35, 40, 46 and 52 are passed in units of basic measures such as Ohm, Volt, Farad and Henry.

Finally, **Figure 16** illustrates the waveform simulation Buck converter in SystemVision environment and this can be compared to the simulation results for the same stimulus obtained from the MATLAB/Simulink environment as shown in **Figure 15**. The waveforms of **Figure 15** and **Figure 16** represent the expected behavior model for the proposed converter.

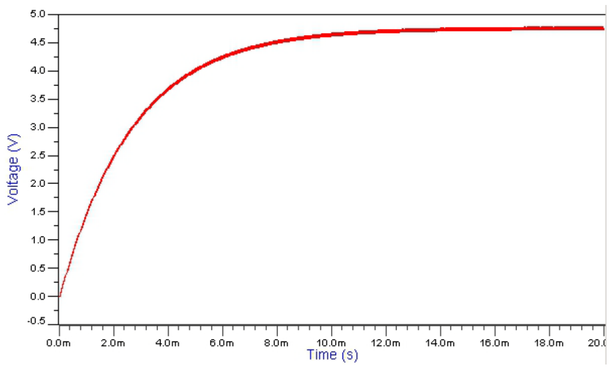


FIGURE 16 WAVEFORM SIMULATION OF THE BUCK CONVERTER IN VHDL-AMS EXECUTED IN SYSTEMVISION ENVIRONMENT

Conclusion

This paper has presented an improvement of the methodology for the synthesis of mixed systems proposed in [3] and implemented with the tool presented in [4] and [5]. The systems were initially modeled at a high level of abstraction using a block diagram approach in the MATLAB/Simulink environment and subsequently translated into a description in structural VHDL-AMS characterizing the synthesis procedure described in [7].

With the MATLAB/Simulink environment, is possible to derive hardware description language code for the models in block diagram. However, in this work the HDL code generation feature is not obtained through the use of a standard MATLAB toolkit but through a separate tool described and used in the paper (the MS²SV tool). The proposed methodology becomes more accessible when this kind of automatic synthesis decreases the design and analysis time, and hence project costs.

Another factor that led to the improvement of previous methodology was the possibility of making more general computational tool, so the MS²SV is independent of an internal (and hence fixed) component library without needing to recompile the source code. With the new methodology, the designer can create their own libraries for subsequent synthesis of the model. To analyze the tool using models for a rudder controller and Buck converter for an UAV, a new design library was created for the models. The library was fully referenced by MS²SV and the tool was accurate in the model synthesis, allowing the validation of this improvement. In future work, more functionality will be added to MS²SV through new methodologies, e.g., optimization and testing of the models in question and a complete model for the UAV will be created.

ACKNOWLEDGEMENTS

The authors would like to thanks to National Council for Scientific and Technological Development (CNPq, process: 141744/2010-3 e 307255/2009-3).

REFERENCES

[1] G. P. Gujarath and Y. S. Ma, Generative CAD integration using commom data model, In: *IEEE CONFERENCE ON AUTOMATION SCIENCE AND ENGINEERING*, Tortonto, Canada: [s.n.], 2010. p. 586-591.

[2] M. F. A. Jabal, M. S. M. Rahim, N. Z. S. Othman and Z. Jupri. A comparative study on extraction method of CAD data from CAD drawings. In: *IEEE INTERNATIONAL CONFERENCE ON INFORMATION MANAGEMENT AND ENGINEERING*, Kuala Limpur, Malaysia: [s.n.], 2009, p. 709-713.

[3] A. C. R. Silva, I. A. Grout, R. Jeffrey and T. O'Shea. Generating VHDL-AMS models of digital-to-analogue converters from Matlab / Simulink. In: *THERMAL, MECHANICAL AND MULTI-PHISICS SIMULATION EXPERIMENTS IN MICROELECTRONICS AND MICROSYSTEMS*, London, England: [s.n.], 2007, p. 1-7.

[4] A. C. R. Silva and I. A. Grout. A methodology and a tool to design of mixed-signal technology. In: *ELECTRONIC, ROBOTICS AND AUTOMOTIVE MECHANICS CONFERENCE*, Cuernavaca, Mexico: [s.n.], 2007, p. 164-169.

[5] A. C. R. da Silva and I. A. Grout. MS²SV: Tradução de modelos em Simulink para VHDL-AMS. *IEEE Latin America Transactions*, v. 9, n. 5, p. 1-10, 2011.

[6] M. A. Bauer, M. Sarrafzadeh and F. Somezi. Fundamental CAD algorithms. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, v. 19, n. 12, p. 1449-1475, 2000.

[7] D. D. Gajski and R. H. Kuhn. Introduction new VLSI tools. *IEEE Computer*, v. 16, n. 12, p. 11-14, 1983.

[8] T. Riesgo, Y. Torroja and E. Torre. Design methodologies based on hardware description languages. *IEEE Transactions on Industrial Electronics*, v. 46, n. 1, p. 3-12, 1999.

[9] E. Christen and K. Bakalar. VHDL-AMS: A hardware description language for analog and mixed-signal application. *IEEE Transactions on Circuits and Systems*, v. 46, n. 10, p. 1263-1272, 1999.

[10] L. Wang and T. J. Kazmierski. VHDL-MAS based genetic optimization of fuzzy logic controllers. *The International Journal for Computation and Mathematics in Electrical and Electronics Engineering*, v. 26, n. 2, p. 447-460, 2007.

[11] MATHWORKS (USA) (Ed.). Simulink: Simulation and Model-Based Design. Disponível em: <http://www.mathworks.com/products/sim ulink/>. Acesso em: 7 mar. 2012.

[12] MENTOR GRAPHICS (USA) (Ed.). *System Modeling*: SystemVision. Disponível em: <http://www.mentor.com/products/sm/system_integration_simulation_analysis/systemvision/>. Acesso em: 7 mar. 2012.

[13] P. J. Asheden, G.D Peterson and D. A. Teegarden. Case Study 2: Mixed-Technlogy Focus. In: *The System Designer's Guide to VHDL-AMS. USA: Elsevier Science*, 2003, p. 427-468.

[14] C. Umans, T. Villa and A. L. Sangiovanni-Vicentelli. Complexity of two-level logic minimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. v. 25, n. 10, p. 1230-1246, 2006.

[15] A. P. SU. Application of ESL Synthesis on GSM Edge Algorithm for Base Station. In: *ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE*, 15., 2010, Taipei, Taiwan. Anais.... Taipei, Taiwan: IEEE, 2010. p. 732 - 737.

[16] G. Hamza, A. Zekry and I. Motawie. Implementation of a complete GPS receiver using Simulink. *IEEE Circuits and Systems Magazine*. v. 9, n. 4, p. 43-51, 2009.

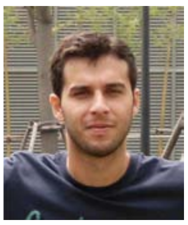
[17] L. Xia, I. M. Bell and A. J. Wilkison. Automated model generation algorithm for high-level fault modeling. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*. v. 29, n. 7, p. 1140-1145, 2010.

[18] I. S. Correa, L. C. Freitas, A. Klautau and J. C. W. A. Costa. VHDL Implementation of a Flexible and Synthesizable FFT Processor. *IEEE Latin America Transaction*, São Paulo, Brasil, v. 10, n. 1, p.1180-1183, 2012.

[19] N. Kapre and A. Dehon. SPICE^2: Spatial Processors Interconnected for Concurrent Execution for Accelerating the SPICE Circuit Simulator Using an FPGA. *IEEE Transactions On Computer-aided Design Of Integrated Circuits And Systems*, USA, v. 31, n. 1, p.9-22, 2012.

[20] Andreas Gerstlauer, Christian Haubelt, Andy D. Pimentel, Todor P. Stefanov, Daniel D. Gajski and Jürgen Teich. Electronic System-Level Synthesis Methodologies. *IEEE Transactions On Computer-aided Design Of Integrated Circuits And Systems*, Austin, USA, v. 28, n. 10, p.1517-1530, 2009.

Author Introduction



Tiago da Silva Almeida was born in Aparecida do Taboado, MS, Brazil, in 1986. He received the B. Sc. degree in Information System from the Centro Universitário de Jales, Jales, São Paulo State and the Msc. in Electrical Engineering from the Universidade Estadual Paulista (UNESP), Ilha Solteira, SP, Brazil, in 2007 and 2009, respectively. Actually, he is student of Ph. D. in the Universidade Estadual Paulista (UNESP) and realizes research in computational methods of synthesis of circuits in high level abstraction, automatic process of measurement and standardized measurement based in IEEE 1451 standard.

Alexandre César Rodrigues da Silva received the B. Sc. degree in Electrical Engineering from the Universidade de Mogi das Cruzes, Mogi das Cruzes, Brazil, in 1984, the M. Sc. and Ph. D. degrees in Electrical Engineering from the Universidade de Campinas (UNICAMP), Campinas, Brazil, in 1989. He received the degree of Associated Lecture from the Universidade Estadual Paulista (UNESP), Ilha Solteira, Brazil, in 2003. In 2007, he developed postgraduate stage at the University of Limerick, Limerick, Ireland. He is currently full Professor - MS6 in the Universidade Estadual Paulista. He has experience in

Electrical Engineering with emphasis in Electronic Circuits and Development Tools for Synthesis and Mixed Systems Digital System (Digital - Analog), acting on the following topics: Smart Sensor Networks IEEE 1451, system (FPGA - DSP - Microcontrollers) and Synthesis of Digital Circuits. Fellow Research Productivity - PQ - Level 2 - CNPq.

Ian Grout received the B. Eng. (Hons) degree in Electronic Engineering from Lancaster University, UK, in 1991 and a Ph. D. degree in Engineering (The Design of ASICs for Control System Applications) from the Lancaster University, UK, in 1994. Since 1998 he has been a lecturer within the Department of Electronic and Computer Engineering at the University of Limerick, Ireland. His main activity areas are microelectronic circuit design, test engineering and remote laboratory design.